USING VIRTUAL NETWORK ADDRESS INFORMATION DURING COMMUNICATIONS

TECHNICAL FIELD

The following disclosure relates generally to inter-computer communication, and more particularly to communication by a computer that hosts virtual domains or virtual users.

BACKGROUND

Computers communicate with each other for a variety of reasons. For example, a user on one computer may wish to send a message (*e.g.*, an email) to a user on a remote computer or to retrieve a Web page from a remote computer. A client computer system can identify and communicate with millions of other computer systems on the Internet by using a unique network address for each such computer. A common type of network address is a unique numeric identifier called an "IP address" (*e.g.*, 198.81.209.25). Each computer also typically has a unique textual Domain Name System (DNS) domain name network address (*e.g.*, "micron.com" or "comp23.MicronPC.com") that is mapped to that IP address.

When a communication is sent from a client (or "source") computer system to a server (or "destination") computer system, the client computer system typically specifies the IP address of the server computer system (either directly or via a domain name that maps to the IP address) in order to facilitate the routing of the communication to the server computer system. Such communications typically take place using the TCP/IP network communications protocol. For example, common application-level communications protocols such as HTTP, FTP, Telnet, and Simple Mail Transfer Protocol (SMTP) typically use TCP/IP as an underlying communications mechanism. The TCP/IP protocol is described in greater detail in "TCP/IP Network Administration,

Second Edition" by Craig Hunt, 1992, O'Reilly & Associates Publishing, which is hereby incorporated by reference in its entirety.

Application programs executing on different computers often communicate by using communication mechanisms provided by the operating systems on their computers. For example, application programs commonly exchange information using TCP/IP communication sockets that are provided by operating systems such as Unix or Microsoft Windows. A socket connection can be described with four pieces of information, those being a source computer IP address and software port and a destination computer IP address and software port. Such a socket connection is typically initiated by an application program executing on a server computer. In particular, a server application creates a socket on the server (*e.g.*, via the "socket" function), binds the server computer IP address and the software port to the socket (*e.g.*, via the "bind" function), and obtains a software port on the server which is mapped to the server application by the server's operating system. The server application then listens for a connection request from a client computer (*e.g.*, via the "listen" function) that includes a source IP address and software port to which the server application can respond.

When a client application program wants to communicate with such a server application, the client application creates a socket on the client (via the "socket" function) and may determine a client computer software port that is to be mapped to the client application. The client application then specifies that the created socket has a destination IP address corresponding to the destination computer and a destination software port that corresponds to the port mapped to the server application program, and uses the socket to make a connection request to the server application (*e.g.*, via the "connect" function). In particular, the connection request is transmitted to the destination IP address and software port, along with the source IP address and software port to allow the server to respond. The inclusion of the source IP address with the connection request can be performed by the client computer operating system, which determines the IP address for the computer on which the client application is executing (*e.g.*, by using the routing table or some other mechanism on the client computer).

The connection request that is sent to the destination IP address and software port is routed to the executing server application, and the application program on the server computer can then accept the connection request (*e.g.*, via the "accept" function) and use the included source IP address and software port to respond to the client application. After the connection is accepted, the two programs can establish a shared communications standard and exchange information. TCP/IP sockets are described in greater detail in "Pocket Guide to TCP/IP Sockets (C Version)" by Kenneth L. Calvert and Michael J. Donaho, 2000, Morgan Kaufmann Publishers, which is hereby incorporated by reference in its entirety.

The Sendmail mail transport agent program is one example of software that communicates in the manner described above. For example, consider a situation in which a user on a client computer creates an email message using a mail user agent program (*e.g.*, elm or Zmail) and specifies that a user on a remote server computer is to be a recipient of the email. The client mail user agent program begins the transfer of the created email to the remote recipient by forwarding the email to a local mail transport agent program (*e.g.*, Sendmail or Zmailer) that is executing on the client computer. A copy of the mail transport agent may already be executing, or the mail user agent may instead invoke the mail transport agent. It is the responsibility of the local mail transport agent to transfer the created email to the remote server computer. If the local mail transport agent is the Sendmail program, it next establishes a socket-based connection with a Sendmail mail transport agent program that is executing on the remote server computer, and sends the email to the server. In some situations, the Sendmail mail transport agent on the client computer will temporarily store the email in an outgoing email queue on the client computer before transferring the email to the server computer. After the mail transport agent on the server computer receives the email, it makes the email available to the remote user recipient, who can use a mail user agent program on the server computer to read the email. The Sendmail program is available at the time of this writing at "http://www.sendmail.org/", and additional details about Sendmail are available in "sendmail, Second Edition" by Bryan Costales with Eric Allman, 1997,

O'Reilly & Associates Publishing, which is hereby incorporated by reference in its entirety.

Figures 1A-1E provide an illustrative example of email communications using the Sendmail program. In particular, Figure 1A illustrates four computing devices that are accessible to each other over a network 190, with each of the computing devices using the Sendmail program (not shown) as their mail transport agent. Each computing device has a distinct assigned IP address 104 and a distinct DNS domain name 102 that is mapped to the assigned IP address. Each of the computing devices also have various users that have access to the computing device (*e.g.*, by having user accounts on the device), with computing devices 100 and 130 each illustrating a list of defined users 106. In the illustrated embodiment, user a1 on computing device 100 will send an email to remote user x1 on computing device 130.

Figure 1B illustrates an example email that is generated by user a1 using a local mail user agent program on computing device 100. Each email consists of a header portion 140 and a body portion 150, with the header portion including various information about the email and the body portion including the contents of the email. The illustrated email header currently includes 4 header lines 141-144 with header names "Date", "From", "To", and "Subject". Those skilled in the art will appreciate that a variety of other header lines can optionally be present.

Figure 1C illustrates the same email after it is transferred to the local Sendmail program executing on computing device 100 for transmittal to computing device 130. As is illustrated, the local Sendmail program adds header lines 145 and 146 to the email message. Header line 145 indicates the sending user and the domain name of the sending computer, and header line 146 includes a unique message identifier for the email. In other situations, the local mail user agent may have already added these header lines when the email was generated, and if so the Sendmail program would not add them.

Figure 1D illustrates the email as it is modified by the remote Sendmail program executing on computing device 130. In particular, when the remote Sendmail program receives the email, it adds a header line 147 to the email that indicates

information about where the email was received from and when the email was received. In the illustrated embodiment, the header line 147 includes both the domain name (*i.e.*, "abc.com") and IP address (*i.e.*, "216.122.95.01") of the computing device from which the email was received.

Figure 1E illustrates the communication that occurs between the two executing Sendmail programs when transferring the email. The two programs communicate using the SMTP communications protocol over TCP/IP sockets. In the illustrated embodiment, the communications from the local client Sendmail executing on computing device 100 are prefaced with ">>>" characters that are not actually transmitted but are shown here for the purposes of distinguishing these communications from those of the remote server Sendmail executing on the computing device 130. As shown in lines 161 and 162, the client Sendmail provides its domain name as part of the connection sequence, and the server Sendmail responds with a confirmation of the domain name identification.

Unfortunately, problems can arise when one computing device must communicate with other computing devices on behalf of a third-party. One reason that such problems can arise stems from the existence of malicious users that intentionally attempt to disguise the identity of themselves and their computing devices in a variety of ways, such as by having their computing devices transmit false domain name or IP address information for identification purposes (referred to as "IP spoofing"). In order to combat such malicious users, many computing devices and application programs have incorporated security measures to attempt to detect and/or prevent such malicious users and their computing devices from establishing connections. In particular, such security measures attempt to verify in various ways that the identification information provided by client computers is accurate. However, such security measures may also detect and/or prevent situations in which a computing device is legitimately acting on behalf of a third-party, such as when the computing device provides identification information that corresponds to the third-party.

As an example of a problem that arises when a computing device attempts to legitimately communicate with other computing devices on behalf of a third-party, consider the illustrative situation shown in Figure 2A. In this situation, computing device 100 is acting as a shared host for four virtual computing devices ("VCDs") 200, 210, 220, and 230, also referred to as "virtual machines" or "virtual computers". Each VCD has distinct virtual network address information, including a distinct DNS domain name 202 and a distinct virtual IP address 204, as well as one or more defined virtual users 206. As with other domain names, the domain name of a VCD is mapped to the virtual IP address of that VCD (*e.g.*, "bcd.com" is mapped to "216.122.95.70"). An owner of a computing device may allow it to act as a host for multiple VCDs or virtual IP addresses for a variety of reasons, such as if the hosting is a service that the owner provides to the domain name or VCD owners for a fee.

While the VCDs are not actual physical devices, they do share the resources of computing device 100 (*e.g.*, memory, storage, and processing power) and can appear to their users as if they were physical devices. For example, user b1 of VCD 200 may be physically using the I/O devices of computing device 110, but may remotely login to VCD 200 (*e.g.*, by using a Telnet program or Web browser) and gain access to the resources of VCD 200. Once access has been gained, user b1 can typically then perform the same types of actions as non-virtual users (*e.g.*, user a1) of computing device 100, such as executing programs or modifying the contents of VCD 200's storage (*e.g.*, storage of computing device 100 that is allocated to VCD 200). In addition to receiving access to computing device 100 resources, the virtual IP addresses of the VCDs are mapped to computing device 100 so that communications sent to those virtual IP addresses will be forwarded to the device in the same manner as communications sent to the device's actual IP address of "216.122.95.01".

As with non-virtual computing devices, the VCDs may need to communicate with other computing devices. For example, user b1 of VCD 200 may wish to send email to user x1 of computing device 130. In order to do so, however, the physical resources of computing device 100 will need to be used, with computing device

100 acting on behalf of VCD 200. Unfortunately, as indicated above, security measures employed by other computing devices or application program may detect and/or prevent such communication by computing device 100 despite the fact that it is acting legitimately on behalf of the third-party VCD 200.

Figures 2B-2E illustrate an example of virtual user b1 of VCD 200 sending email to user x1 of computing device 130 using the Sendmail program as the mail transport agent, and of problems caused by security measures employed by the Sendmail program (which is responsible in this example for the inter-computer communication). In particular, Figure 2B illustrates an example email that is generated by user b1 using a local mail user agent program on VCD 200. This email is similar to that illustrated in Figure 1B, with the exception that header line 242 identifies the sender as being a user of the "bcd.com" domain name for VCD 200 rather than the "abc.com" domain name for computing device 100. Figure 2C illustrates the email after it is transferred to a local Sendmail program executing on VCD 200, such as by invoking the program. As with Figure 1C, the local Sendmail program adds header lines 245 and 246 to the email message, but the header lines in this example also use the "bcd.com" domain name rather than computing device 100's "abc.com" domain name.

Figure 2D illustrates the email as it is modified by the remote Sendmail program executing on computing device 130. In particular, the remote Sendmail program adds a header line 247 to the email that indicates information about where the sender of the email. However, as is illustrated in Figure 2D (in bold for the sake of clarity), the remote server Sendmail identifies the domain name and IP address of the sending computing device to be that of the physical device used for the sending (*i.e.*, "abc.com" and "216.122.95.01") rather than those of the VCD which was responsible for sending the email. The communication that occurs between the two executing Sendmail programs, shown in Figure 2E, illustrates the same problem. In particular, as shown in line 261, the client Sendmail provides an indication that its domain name is "bcd.com". The server Sendmail believes the actual domain name of the client to be "abc.com", however, based on examining the IP address from which the communication request was

made and mapping the IP address to its domain name. The server Sendmail indicates the inconsistency between the indicated domain name of the client and what it believes to be the true domain name in line 262 by including the true domain name in parentheses (illustrated in bold for the sake of clarity). The server Sendmail also adds an entry to an Error file indicating that the host "abc.com" claimed to be "bcd.com".

The misidentification of the email sender in the illustrated situation is caused by Sendmail's use of the standard socket network communications mechanism. In particular, even if the Sendmail program is invoked by another program executing in VCD 200 such as a mail user agent, the Sendmail program needs to have unrestricted access to various resources of computing device 100 and thus needs to execute with the highest level of system administrator privileges for computing device 100 (*e.g.*, the "root" user for a Unix system). When the operating system of computing device 100 determines the network address information that corresponds to the socket created by the Sendmail program, it determines that the Sendmail program is executing for a user of computing device 100 having system administrator privileges and it retrieves the network address information for computing device 100. In this manner, the network address information of computing device 100 gets provided for the created socket rather than the network address information of VCD 200. Those skilled in the art will appreciate that a similar problem could occur for various other reasons in other situations. For example, if computing device 100 provided a single executing copy of a network communications program that processed the communications requests for all of the application programs running on the VCDs, that executing program may similarly use the network address information for computing device 100 rather than for any particular user or VCD executing an application program.

Unfortunately, the misidentification of the email sender in the illustrated situation can cause various problems. Depending on the configuration of the remote mail transport agent, such an email may not even be accepted. Moreover, even if the email is accepted and provided to user x1, the email will contain inconsistent information about the sender. In particular, header line 247 indicates that the email came from a user at a

device with a domain name of "abc.com" and an IP address of 216.122.95.01, while header lines 242, 245 and 246 indicate that the email came from a user at a device with a domain name of "bcd.com" and an IP address of 216.122.95.70. Such inconsistent information can cause the recipient of the email to distrust the email or refuse to accept it. For situations in which a company is providing hosting services to customers, such problems can cause loss of business and other problems. Moreover, those skilled in the art will appreciate that these problems are not limited to email messages, and can instead occur for any type of inter-computer communication.

Thus, a need exists for a computing device that is acting legitimately on behalf of third-parties (*e.g.*, a computing device acting as a host to VCDs or to virtual network addresses) to be able to communicate with other computing devices for those third-parties without triggering security measures employed by those other computing devices.

BRIEF DESCRIPTION OF THE DRAWINGS

Figures 1A and 2A are network diagrams illustrating interconnected network devices.

Figures 1B-1E and 2B-2E are examples of email messages communicated between network devices.

Figures 3A and 3B are examples of email messages communicated between network devices using techniques of the present invention.

Figure 4 is a block diagram illustrating an embodiment of the present invention.

Figure 5 is a flow diagram of an embodiment of the Create/Store Message For Virtual User routine.

Figure 6 is a flow diagram of an embodiment of the Send Messages For Virtual User routine.

DETAILED DESCRIPTION

A software facility is described below that employs virtual network address information in communications with other computing devices or software programs. In some embodiments, the facility is used by a computing device that hosts multiple virtual domains each having a distinct domain name and having one or more virtual users of the domain. In such embodiments, the host computing device communicates with others on behalf of the virtual domains or virtual users by employing virtual network address information as part of the communications.

For illustrative purposes, some embodiments of the software facility are described below in which email messages are exchanged using the Sendmail message transport agent program, and in which a computing device that host multiple virtual domains communicates with other computers on behalf of virtual users of those domains. However, those skilled in the art will appreciate that the techniques of the facility can be used in a wide variety of other situations, some of which are discussed below, and that the use of the facility is not limited to the exchange of email messages, to the use of the Sendmail program, or to situations in which a computing device is hosting virtual domains or virtual users.

In particular, Figure 4 illustrates a virtual domain hosting computer system 400 suitable for executing the software facility, various customer computer systems 450 from which users can access the hosting computer system, and various other computer systems 470 that can communicate with the hosting computer system. The hosting computer system 400 includes a CPU 405, various I/O devices 410, storage 420 (*e.g.*, a hard drive), and memory 430. The I/O devices include a display 411, a network connection 412, a computer-readable media drive 413, and other I/O devices 415.

The storage includes groups of information for multiple virtual domains 440 (or VCDs to which the virtual domains correspond) that are hosted by the hosting computer system. Each virtual domain can contain various information for use by users of the domain, such as a mail agent program 442 for creating and storing email messages

and other domain-specific software and/or data 448. Each virtual domain also includes configuration information 446 for the domain (*e.g.*, a virtual domain name for the domain, a virtual IP address to which the virtual domain name is mapped, a list of users of the virtual domain, etc.), and an outgoing mail queue in which mail created by users of the virtual domain for recipients at other computing devices is temporarily stored before transfer. In some embodiments, the mail agent program 442 will store the outgoing email in the virtual domain queue, while in other embodiments the mail agent program will transfer the email to a mail transport agent that will perform the storage. Those skilled in the art will appreciate that a variety of other data and software could similarly be stored for each virtual domain, and that one or more of the virtual domains may alternately lack some of the displayed information (*e.g.*, there may be a single stored mail user agent that the users in all of the virtual domains use). Those skilled in the art will also appreciate that in some embodiments different virtual domains can use different mail agent programs 442.

In addition to the virtual domains, the storage also includes configuration information 424 for the computer system (*e.g.*, information about the various virtual users and virtual domains, such as the locations in which the virtual domain information is stored) and an outgoing mail queue in which mail created by users of the computer system that are not users of any of the virtual domains is temporarily stored before transfer. The storage also includes a Virtual Domain Mail Transport Agent ("VDMTA") program 422 that, when executed, will transmit the email messages for the virtual domains to remote computer systems as appropriate. In some embodiments the VDMTA may be invoked by a mail agent 442 for a virtual domain when mail is available in the outgoing mail queue for the virtual domain, while in other embodiments the VDMTA periodically (*e.g.*, every 30 minutes) checks the outgoing mail queues for the various virtual domains.

Executing copies of one or more mail user/storage agents 434 and one or more VDMTAs 432 are present in the memory. In some embodiments, a single copy of the VDMTA executes and transfers mail for all of the virtual domains, and in other

embodiments multiple copies of the VDMTA can be executing for different virtual domains (*e.g.*, when each virtual domain invokes a copy of the VDMTA when there is mail to be sent from the virtual domain's outgoing mail queue).

When processing the email for a virtual domain, an executing VDMTA retrieves the mail stored in the outgoing mail queue for the virtual domain and retrieves the virtual network address information for the virtual domain (*e.g.*, from the stored domain configuration information for the virtual domain). For each email message in the queue, the VDMTA then determines any recipients on remote computing devices, establishes a connection with those remote computing devices on behalf of the virtual domain by using the retrieved virtual network address information, and then transmits the email to the remote computing devices. In particular, before establishing a connection with a remote computing device, the VDMTA binds the virtual network address information to the communication mechanism to be used in such a manner that the virtual network address information is used in place of that of the hosting computer system. For example, in the illustrated embodiment in which TCP/IP sockets are used, the virtual IP address for the virtual domain is bound to the socket before a connection request is sent to the remote computer.

If the hosting computer system has its own defined users that are not virtual users, such users may also create email messages for recipients of other computer systems. If so, they will use a mail transport agent (*e.g.*, VDMTA) to send the email messages to those other computer systems using the network address information of the hosting computer system. In some circumstances, such as when the mail transport agent cannot immediately transmit an email message, a message will be stored in the outgoing mail queue 426 for the computer system. When a mail transport agent later retrieves the mail from the mail queue 426 for transmittal, the mail transport agent will send the email messages using the hosting computer system network address information.

The various virtual domain components and information on the hosting computer system may be accessed by users and software in a variety of ways. For example, some users may have physical access to the hosting computer system, and may

thus be able to gain access via the I/O devices 410. Alternately, other users can use the I/O devices 454 and software (*e.g.*, a Telnet program 459 executing in memory 457) that are provided by one of the consumer computer systems to remotely access a hosted virtual domain (*e.g.*, via the Internet and/or the World Wide Web). In addition to user accesses of the virtual domain functionality, the hosting computer system can also exchange information (*e.g.*, email messages) with various remote server computer systems 470, such as via mail transport agents 479 executing in memory 477 of the server computers.

Those skilled in the art will appreciate that computer systems 400, 450, and 470 are merely illustrative and are not intended to limit the scope of the present invention. Computer system 400 may be connected to other devices that are not illustrated, including through one or more networks such as the Internet or via the World Wide Web (WWW). In addition, the functionality provided by the illustrated components may in some embodiments be combined in fewer components or distributed in additional components. Similarly, in some embodiments the functionality of some of the illustrated components may not be provided and/or other additional functionality may be available. For example, other components could additionally be available to perform administrative functions for hosting customers (*e.g.*, establishing new virtual domains and providing account status information for existing customers) and to obtain new virtual network address information (*e.g.*, domain names and virtual IP addresses) for new customers.

Those skilled in the art will also appreciate that, while various components are illustrated as being in memory or on storage, these items or portions of them can be transferred between memory and other storage for purposes of memory management and data integrity. The software components and data structures may also be stored as instructions on a computer-readable medium (*e.g.*, a hard disk, a memory, or a portable article to be read by an appropriate drive), and transmitted as generated data signals on a variety of computer-readable transmission mediums, including wireless-based and

wired/cable-based mediums. Accordingly, the present invention may be practiced with other computer system configurations.

Moreover, those skilled in the art will appreciate that the software facility can be used in various environments other than the Internet. In addition, a hosting computer system may comprise any combination of hardware or software that can provide hosting functionality. Similarly, a customer system may comprise any combination of hardware or software that can interact with the hosting computer system. These systems may include television-based systems or various other consumer products. The various computer systems can also operate on a wide variety of operating system types (*e.g.*, Windows, Linux, Unix, MacOS, BEOS, PalmOS, EPOC, Windows CE, FLEXOS, OS/9, JavaOS, etc.), and need not share the same operating system.

Figures 3A and 3B illustrate using the software facility to perform the same transmission of an email message as was previously discussed with respect to Figures 2A-2E, that being from virtual user b1 of VCD 200 to user x1 of computing device 130. In this example, computing device 100 illustrated in Figure 2A is a virtual hosting computer system 400 as illustrated in Figure 4, with the virtual bcd.com domain corresponding to VCD 200 being one of the virtual domains 440, and remote computing device 130 in Figure 2A is one of the server computers 470 illustrated in Figure 4. Thus, the email to be sent is created by user b1 of virtual domain bcd.com using a mail/user storage agent 442 available on the hosting computer system, and is then temporarily stored in the outgoing mail queue 444 for the bcd.com virtual domain. A Sendmail program modified to act as a VDMTA is executing on the hosting computer system to retrieve the email from virtual domain bcd.com's outgoing mail queue 444 (as well as any other email present in the queue), use the virtual network address information for the bcd.com domain to establish connections with remote computing device 130 on behalf of the bcd.com virtual domain, and then transmit the email to the remote computing device.

Figure 3A illustrates the email as it is modified by the remote Sendmail program executing on remote computing device 130. In particular, the remote Sendmail program adds a header line 347 to the email that indicates information about the identity

of the email sender. However, with the techniques of the software facility used by the local VDMTA Sendmail program, the remote Sendmail program receives the virtual network address information for virtual domain bcd.com rather than the network address information for computing device 100. As a result, the remote server Sendmail identifies the domain name and IP address of the sending computing device to be that of VCD 200 (*i.e.*, "bcd.com" and "216.122.95.70") in line 347, as is illustrated in bold for the sake of clarity. In a similar manner, the communication that occurs between the two executing Sendmail programs reflects the virtual network address information, as illustrated in Figure 3B. In particular, as shown in line 361, the client VDMTA Sendmail provides its domain name as "bcd.com". Since the remote server Sendmail believes the actual domain name of the client to be "bcd.com", line 362 (illustrated in bold for the sake of clarity) indicates that it believes the remote device to be "bcd.com" (since no other identification information is added in parentheses). In addition, the server Sendmail on computing device 130 does not add an entry to an Error file since it did not detect any errors.

Those skilled in the art will appreciate that an existing mail transport agent can be modified to act as a VDMTA in a variety of ways, and that the modifications will be specific to the particular mail transport agent used. Tables 1-4 below illustrate examples of modifications that can be made to a Sendmail version 8.9.3 program so that it acts as a VDMTA.

In particular, Table 1 illustrates C-language instructions that the modified client Sendmail can execute before making a connection with a remote server Sendmail so that the modified client Sendmail will use appropriate virtual network address information in place of the actual network address information for the computing device on which the modified client Sendmail is executing. These instructions can be added, for example, in the "makeconnection" procedure in the daemon.c file after the "socket()" system call and before the "connect()" system call.

```
{
register struct hostent *hp;
struct sockaddr_in psa;
char *vhost, vhostaddr[50];
```

```
        bzero((char *)&psa, sizeof(struct sockaddr_in));
        psa.sin_family=AF_INET;

        vhost = macvalue ('j', e);
        hp = sm_gethostbyname(vhost);
        if (hp == NULL)
                syserr("544 host \"%s\" unknown", vhost);
        else {
                strcpy(vhostaddr, inet_ntoa(*(struct in_addr *)hp->h_addr));
                psa.sin_addr.s_addr = inet_addr(vhostaddr);

/*                      bcopy(hp->h_addr, &DaemonAddr.sin.sin_addr,INADDRSZ);
*/
        }

        psa.sin_port=htons(INADDR_ANY);
        if (bind(s, (struct sockaddr *)&psa, sizeof(struct sockaddr_in)) <0 {
                printf("Error binding. \n");
        }
        }
```

Table 1

Table 2 illustrates C-language instructions that the modified client Sendmail can execute so that created emails are stored in an outgoing mail queue for the appropriate virtual domain or machine before they are transmitted. These instructions can be added, for example, in the "main" procedure in the main.c file after the uid and gid are determined and before "save command line arguments" and the reading of the configuration files.

```
        {
#define VIRTUAL_UIDMIN 1000
#define VIRTUAL_UIDMAX 30000

                struct passwd *pp;
                login_cap_t *lcptr;

                if ((RealUid!=0) && (geteuid() == 0) &&
                (RealUid >= VIRTUAL_UIDMIN) && (RealUid <= VIRTUAL_UIDMAX))
                {
                    /* the command is invoked from a user that controls a
                     * virtual domain, as a setuid binary. Do chroot to
                     * the virtual domain. */
                   pp = sm_getpwuid(ReadUid);
                   lcptr = login_getclass(pp->pw_class);
                   setusercontext(lcptr, pp, pp->pw_uid, LOGIN_SETRESOURCES);
```

```
            chroot(pp->pw_dir);
            chdir("/");
            setegid(pp->pw_gid);
            seteuid(pp->pw_uid);
            RunAsUid = RealUid;
            RunAsGid = RealGid;
        }
    }
```

Table 2

Table 3 illustrates a Perl-language script named "vsmq.pl" that when executed will process the queues of outgoing stored emails for the various virtual users. In the illustrated example, each virtual user X has a home directory at "/usr/home/X/" and has a outgoing mail queue at "/usr/home/X/var/spool/mqueue". The script can be stored in any location from which it can be executed, such as "/usr/local/bin/" on some Unix-based computing devices.

```
#!/usr/local/bin/perl
# Flush mail queue directory for each virtual domain.
#

$vsmcmd = '/usr/local/bin/virtual /bin/sendmail –q -v';
$conf = '/etc/ip.conf';

open IPCONF, $conf\\ die "Could not open $conf\n";
while (<IPCONF>) {
    chop;
    ($ip, $vuser) = split /\s+/;
    if (defined $vusers) {
        $vusers = join ' ', $vusers, $vuser;
    } else {
        $vusers = $vuser;
    }
}
close(IPCONF);
#print "$vusers \n";

foreach $user (split / /,$vusers) {
    $dir = "/usr/home/$user";
```

```
        $qdir = "${dir}/var/spool/mqueue";
        if ((-d $qdir) and (-e <${qdir}/qf*>)){
#           print "$qdir \n";
          system('/usr/bin/su',$user,'-c',$vsmcmd);
        }
}
```

Table 3

Finally, the following line can be added to the Unix root user's crontab so that the script illustrated in Table 3 will be automatically executed.

```
40 * * * * /usr/local/bin/vsmq.pl >>\& /var/log/maillog
```

Those skilled in the art will appreciate that the locations of the files may vary on different computing devices and with different operating systems.

Figure 5 is a flow diagram of an embodiment of the Create/Store Message For Virtual User routine 500. The routine creates email messages for a virtual user of a hosted virtual domain, and stores the created email messages in the outgoing message mail queue for the virtual domain for later transmittal. Those skilled in the art will appreciate that in other embodiments an email creation routine may provide a created email to a separate routine for storage in the outgoing mail queue (*e.g.*, a mail transport agent), or that in other embodiments the email creation routine may invoke a mail transport agent (*e.g.*, periodically or after the creation of some or all email messages) to transmit created messages.

The routine begins in step 505 where an indication to create a message is received. The routine continues to step 510 where it receives an indication of the contents of the message, of the message recipients, and optionally of other message-related information such as a message Subject. In step 515 the routine then determines the virtual user name of the sender, the virtual network address information of the virtual user's domain, and the current time. The determined information is used in step 520 to add various headers to the created message. In step 525, the created message is added to the outgoing message queue for the virtual domain (or the virtual computer system supporting the virtual domain). In step 530, it is determined whether there are more

messages to be created and stored, and if so returns to step 505. If not, the routine continues to step 595 and ends.

Figure 6 is a flow diagram of an embodiment of the Send Messages For Virtual User routine 600. The routine retrieves the messages stored in the outgoing mail queue of a virtual domain, and sends the messages to the appropriate remote computing devices using the virtual network address information of the virtual domain. The routine begins in step 605 where it receives an indication of a virtual domain whose queued outgoing messages are to be sent. The routine continues to step 610 where it retrieves the queued outgoing messages for the indicated virtual domain, and then continues to step 612 to retrieve the virtual network address information for the indicated virtual domain. The routine continues to step 615 to select the next of the retrieved messages, beginning with the first. In step 620, the routine then determines the destination computing devices of the message (*e.g.*, from the message headers), and continues to step 625 to select the next destination device, beginning with the first. In step 630, the routine then determines if the selected computing device is a remote device.

If it is determined that the selected computing device is not remote, then at least one of the recipients of the message is a virtual user of the same virtual domain as that of the message sender, and the routine continues to step 640. In step 640, the routine optionally adds headers to the message using the virtual network address information of the virtual domain (*e.g.*, to add mandatory headers not included by the mail creation agent). In step 645, the routine then sends the message to the local recipients (*e.g.*, by adding a copy of the message to incoming mail queues for each of the recipients).

If it was instead determined that the selected computing device is remote, then the routine continues to step 655 to create a communication socket to be used for transmitting the email to the selected computing device. The routine then continues to step 660 to bind the virtual network address information (*e.g.*, a virtual IP address) to the socket. In step 665, the routine then uses the socket to make a connection to the IP address of the remote machine (which can be obtained from the email message or from the domain name of the remote device that is included in the email message), using the

virtual network address information as the address of the sender. The routine then continues to step 670 to optionally add headers to the message using the virtual network address information of the virtual domain, and then sends the message to the remote recipients over the socket.

After steps 645 or 675, the routine continues to step 680 to determine if there are more destination machines for the selected email message. If so, the routine returns to step 625, and if not continues to step 685 to determines if there are more messages from the queue to be sent. If so, the routine returns to step 615, and if not the routine continues to step 690 to determine if there are more virtual domains whose messages are to be sent. If so, the routine returns to step 605, and if not the routine continues to step 695 and ends.

Those skilled in the art will also appreciate that in some embodiments the functionality provided by the routines discussed above may be provided in alternate ways, such as being split among more routines or consolidated into less routines. Similarly, in some embodiments illustrated routines may provide more or less functionality than is described, such as when other illustrated routines instead lack or include such functionality respectively, or when the amount of functionality that is provided is altered.

From the foregoing it will be appreciated that, although specific embodiments have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims. In addition, while certain aspects of the invention are presented below in certain claim forms, the inventors contemplate the various aspects of the invention in any available claim form. For example, while only one some aspects of the invention may currently be recited as being embodied in a computer-readable medium, other aspects may likewise be so embodied. Accordingly, the inventors reserve the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the invention.